Importing Libraries

In [16]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [17]:
```python
df=pd.read_csv("innovation_and_development_database.csv")
df
```

Out[17]:

	country	code	year	eap	eca	lac	mena	sha	sa	hi	...	y	stockpatEPO	
0	Aruba	ABW	1960	0	0	1	0	0	0	0.0	...	NaN	NaN	
1	Aruba	ABW	1961	0	0	1	0	0	0	0.0	...	NaN	NaN	
2	Aruba	ABW	1962	0	0	1	0	0	0	0.0	...	NaN	NaN	
3	Aruba	ABW	1963	0	0	1	0	0	0	0.0	...	NaN	NaN	
4	Aruba	ABW	1964	0	0	1	0	0	0	0.0	...	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8290	Zimbabwe	ZWE	1998	0	0	0	0	1	0	0.0	...	8.290000e+09	8.0	12
8291	Zimbabwe	ZWE	1999	0	0	0	0	1	0	0.0	...	8.230000e+09	8.0	12
8292	Zimbabwe	ZWE	2000	0	0	0	0	1	0	0.0	...	7.830000e+09	8.0	12
8293	Zimbabwe	ZWE	2001	0	0	0	0	1	0	0.0	...	NaN	8.0	12
8294	Zimbabwe	ZWE	2002	0	0	0	0	1	0	0.0	...	NaN	NaN	

8295 rows × 33 columns

Data Cleaning and Data Preprocessing

In [18]:
```python
df=df.fillna(1)
df
```

Out[18]:

	country	code	year	eap	eca	lac	mena	sha	sa	hi	...	y	stockpatEPO
0	Aruba	ABW	1960	0	0	1	0	0	0	0.0	...	1.000000e+00	1.0
1	Aruba	ABW	1961	0	0	1	0	0	0	0.0	...	1.000000e+00	1.0
2	Aruba	ABW	1962	0	0	1	0	0	0	0.0	...	1.000000e+00	1.0
3	Aruba	ABW	1963	0	0	1	0	0	0	0.0	...	1.000000e+00	1.0
4	Aruba	ABW	1964	0	0	1	0	0	0	0.0	...	1.000000e+00	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...

	country	code	year	eap	eca	lac	mena	sha	sa	hi	...	y	stockpatEPO	
8290	Zimbabwe	ZWE	1998	0	0	0	0	1	0	0.0	...	8.290000e+09	8.0	12
8291	Zimbabwe	ZWE	1999	0	0	0	0	1	0	0.0	...	8.230000e+09	8.0	12
8292	Zimbabwe	ZWE	2000	0	0	0	0	1	0	0.0	...	7.830000e+09	8.0	12
8293	Zimbabwe	ZWE	2001	0	0	0	0	1	0	0.0	...	1.000000e+00	8.0	12
8294	Zimbabwe	ZWE	2002	0	0	0	0	1	0	0.0	...	1.000000e+00	1.0	

8295 rows × 33 columns

In [19]:
```
df.columns
```

Out[19]:
```
Index(['country', 'code', 'year', 'eap', 'eca', 'lac', 'mena', 'sha', 'sa',
       'hi', 'pat', 'patepo', 'royal', 'rdexp', 'rdper', 'rdfinabro',
       'rdfinprod', 'rdperfprod', 'rdperfhe', 'rdperfpub', 'lowrdexp',
       'lowrdfinprod', 'lowrdperfprod', 'y', 'stockpatEPO', 'poptotal',
       'labor', 'rdexpgdp', 'patgrantedstock', 'plantpatstock',
       'designpatstock', 'plantpat', 'designpat'],
      dtype='object')
```

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8295 entries, 0 to 8294
Data columns (total 33 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   country          8295 non-null   object 
 1   code             8295 non-null   object 
 2   year             8295 non-null   int64  
 3   eap              8295 non-null   int64  
 4   eca              8295 non-null   int64  
 5   lac              8295 non-null   int64  
 6   mena             8295 non-null   int64  
 7   sha              8295 non-null   int64  
 8   sa               8295 non-null   int64  
 9   hi               8295 non-null   float64
 10  pat              8295 non-null   float64
 11  patepo           8295 non-null   float64
 12  royal            8295 non-null   float64
 13  rdexp            8295 non-null   float64
 14  rdper            8295 non-null   float64
 15  rdfinabro        8295 non-null   float64
 16  rdfinprod        8295 non-null   float64
 17  rdperfprod       8295 non-null   float64
 18  rdperfhe         8295 non-null   float64
 19  rdperfpub        8295 non-null   float64
 20  lowrdexp         8295 non-null   float64
 21  lowrdfinprod     8295 non-null   float64
 22  lowrdperfprod    8295 non-null   float64
 23  y                8295 non-null   float64
 24  stockpatEPO      8295 non-null   float64
 25  poptotal         8295 non-null   float64
 26  labor            8295 non-null   float64
 27  rdexpgdp         8295 non-null   float64
 28  patgrantedstock  8295 non-null   float64
 29  plantpatstock    8295 non-null   float64
 30  designpatstock   8295 non-null   float64
 31  plantpat         8295 non-null   float64
 32  designpat        8295 non-null   float64
```

```
dtypes: float64(24), int64(7), object(2)
memory usage: 2.1+ MB
```

In [21]:

```
data=df[['year' ,'lac']]
data
```

Out[21]:

	year	lac
0	1960	1
1	1961	1
2	1962	1
3	1963	1
4	1964	1
...	...	...
8290	1998	0
8291	1999	0
8292	2000	0
8293	2001	0
8294	2002	0

8295 rows × 2 columns

In [25]:

```
a=df.head(60)
a
```

Out[25]:

	country	code	year	eap	eca	lac	mena	sha	sa	hi	...	y	stockpatEPO	poptotal	labor
0	Aruba	ABW	1960	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
1	Aruba	ABW	1961	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
2	Aruba	ABW	1962	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
3	Aruba	ABW	1963	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
4	Aruba	ABW	1964	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
5	Aruba	ABW	1965	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
6	Aruba	ABW	1966	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
7	Aruba	ABW	1967	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
8	Aruba	ABW	1968	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
9	Aruba	ABW	1969	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
10	Aruba	ABW	1970	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
11	Aruba	ABW	1971	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
12	Aruba	ABW	1972	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
13	Aruba	ABW	1973	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
14	Aruba	ABW	1974	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
15	Aruba	ABW	1975	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0

	country	code	year	eap	eca	lac	mena	sha	sa	hi	...	y	stockpatEPO	poptotal	labor
16	Aruba	ABW	1976	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
17	Aruba	ABW	1977	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
18	Aruba	ABW	1978	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
19	Aruba	ABW	1979	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
20	Aruba	ABW	1980	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
21	Aruba	ABW	1981	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
22	Aruba	ABW	1982	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
23	Aruba	ABW	1983	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
24	Aruba	ABW	1984	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
25	Aruba	ABW	1985	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
26	Aruba	ABW	1986	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
27	Aruba	ABW	1987	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
28	Aruba	ABW	1988	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
29	Aruba	ABW	1989	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
30	Aruba	ABW	1990	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
31	Aruba	ABW	1991	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
32	Aruba	ABW	1992	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
33	Aruba	ABW	1993	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
34	Aruba	ABW	1994	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
35	Aruba	ABW	1995	0	0	1	0	0	0	0.0	...	1.0	0.0	1.0	1.0
36	Aruba	ABW	1996	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
37	Aruba	ABW	1997	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
38	Aruba	ABW	1998	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
39	Aruba	ABW	1999	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
40	Aruba	ABW	2000	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
41	Aruba	ABW	2002	0	0	1	0	0	0	0.0	...	1.0	1.0	1.0	1.0
42	Andorra	ADO	1960	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
43	Andorra	ADO	1961	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
44	Andorra	ADO	1962	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
45	Andorra	ADO	1963	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
46	Andorra	ADO	1964	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
47	Andorra	ADO	1965	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
48	Andorra	ADO	1966	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
49	Andorra	ADO	1967	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
50	Andorra	ADO	1968	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
51	Andorra	ADO	1969	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0

	country	code	year	eap	eca	lac	mena	sha	sa	hi	...	y	stockpatEPO	poptotal	labor
52	Andorra	ADO	1970	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
53	Andorra	ADO	1971	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
54	Andorra	ADO	1972	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
55	Andorra	ADO	1973	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
56	Andorra	ADO	1974	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
57	Andorra	ADO	1975	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
58	Andorra	ADO	1976	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0
59	Andorra	ADO	1977	0	1	0	0	0	0	0.0	...	1.0	1.0	1.0	1.0

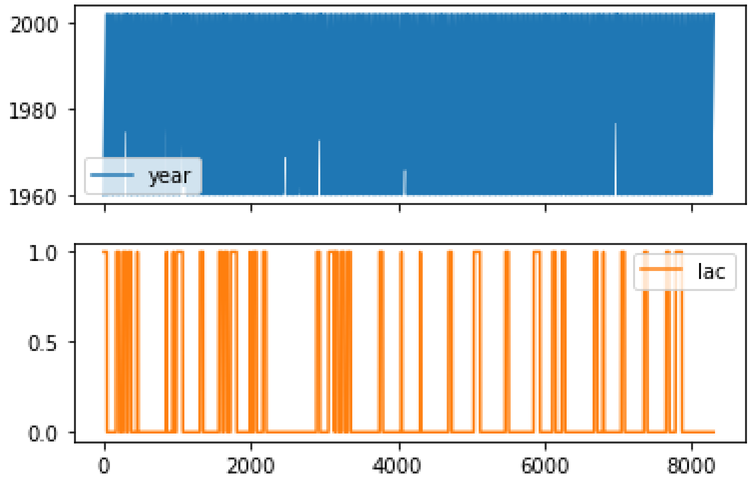
60 rows × 33 columns

Line chart

In [26]:

```
data.plot.line(subplots=True)
```

Out[26]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)

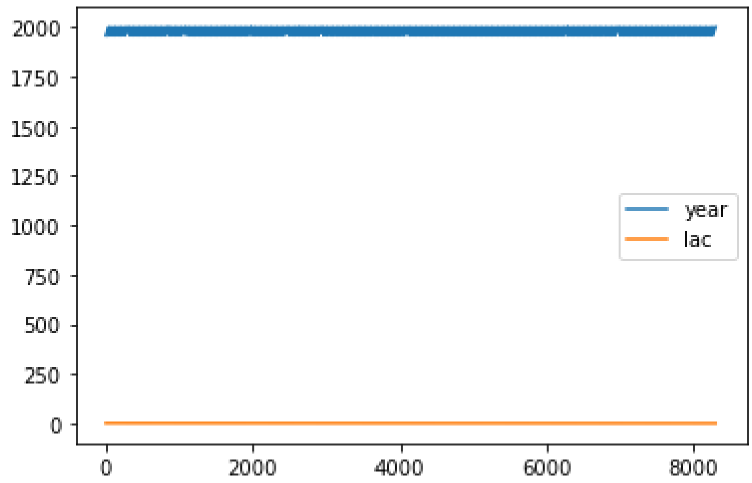


Line chart

In [27]:

```
data.plot.line()
```

Out[27]: <AxesSubplot:>

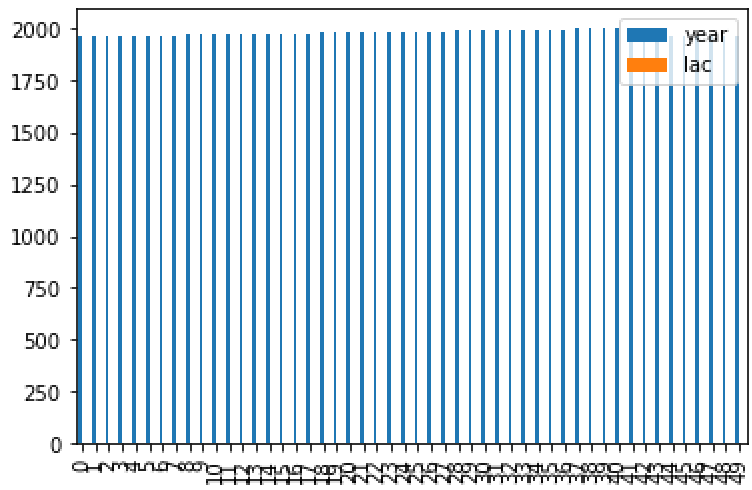


Bar chart

In [28]:
```
b=data[0:50]
```

In [29]:
```
b.plot.bar()
```

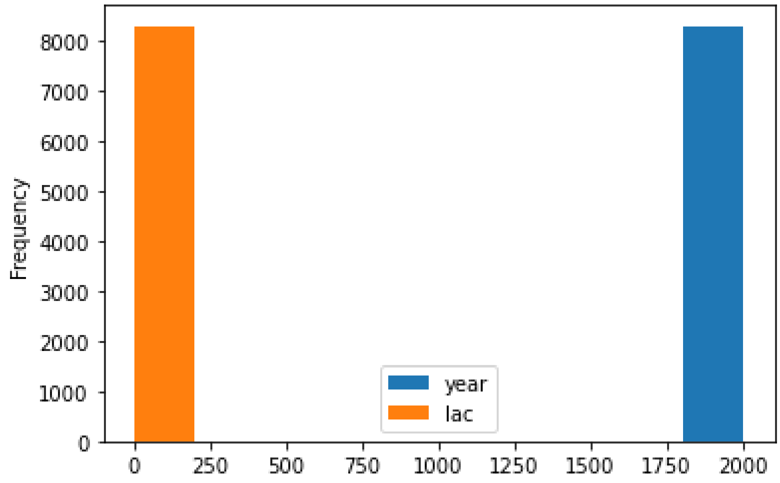
Out[29]: <AxesSubplot:>



Histogram

In [30]:
```
data.plot.hist()
```

Out[30]: <AxesSubplot:ylabel='Frequency'>

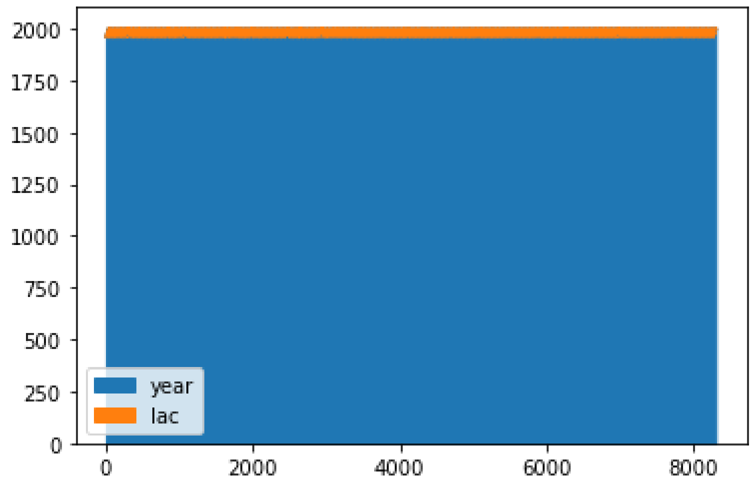


Area chart

In [31]:

```
data.plot.area()
```

Out[31]: <AxesSubplot:>

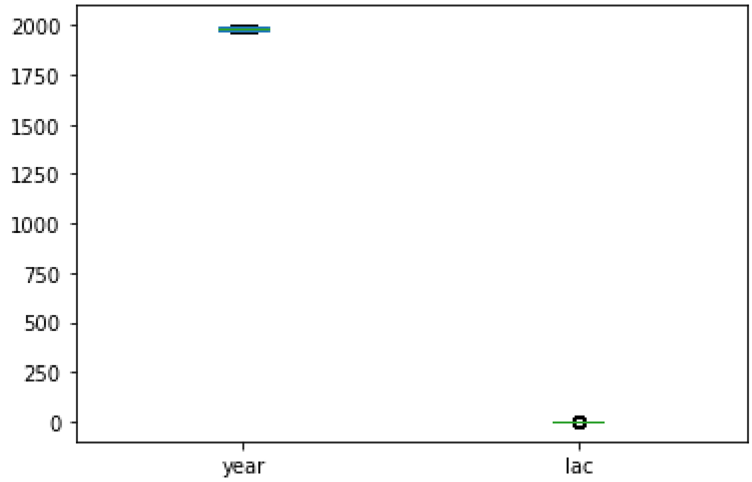


Box chart

In [32]:

```
data.plot.box()
```

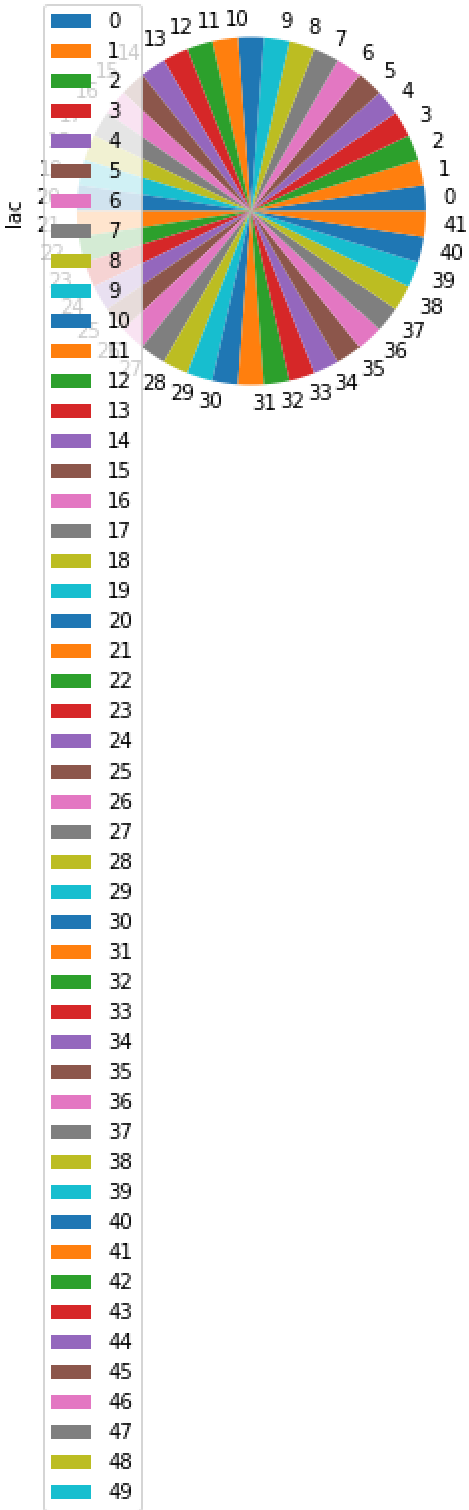
Out[32]: <AxesSubplot:>



Pie chart

In [33]:
```
b.plot.pie(y='lac' )
```

Out[33]: <AxesSubplot:ylabel='lac'>

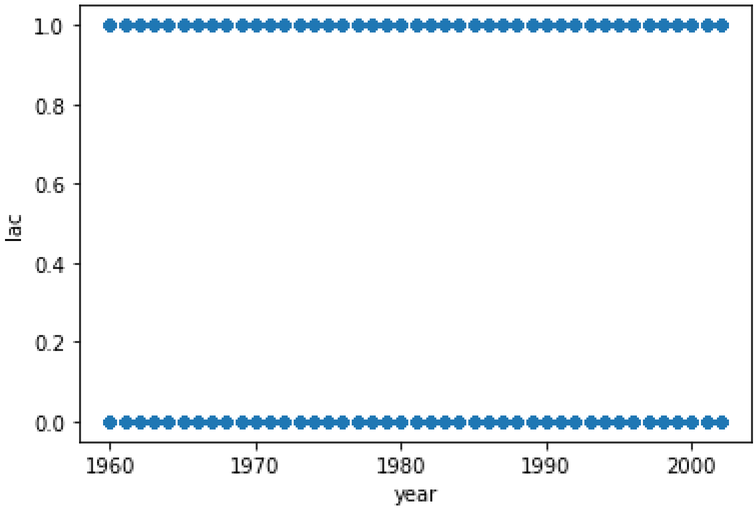


Scatter chart

In [34]:

```
data.plot.scatter(x='year' ,y='lac')
```

Out[34]: <AxesSubplot:xlabel='year', ylabel='lac'>



In [35]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8295 entries, 0 to 8294
Data columns (total 33 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   country          8295 non-null   object 
 1   code             8295 non-null   object 
 2   year             8295 non-null   int64  
 3   eap              8295 non-null   int64  
 4   eca              8295 non-null   int64  
 5   lac              8295 non-null   int64  
 6   mena             8295 non-null   int64  
 7   sha              8295 non-null   int64  
 8   sa               8295 non-null   int64  
 9   hi               8295 non-null   float64
 10  pat              8295 non-null   float64
 11  patepo           8295 non-null   float64
 12  royal            8295 non-null   float64
 13  rdexp            8295 non-null   float64
 14  rdper            8295 non-null   float64
 15  rdfinabro        8295 non-null   float64
 16  rdfinprod        8295 non-null   float64
 17  rdperfprod       8295 non-null   float64
 18  rdperfhe         8295 non-null   float64
 19  rdperfpub        8295 non-null   float64
 20  lowrdexp         8295 non-null   float64
 21  lowrdfinprod     8295 non-null   float64
 22  lowrdperfprod    8295 non-null   float64
 23  y                8295 non-null   float64
 24  stockpatEPO      8295 non-null   float64
 25  poptotal         8295 non-null   float64
 26  labor            8295 non-null   float64
 27  rdexpgdp         8295 non-null   float64
 28  patgrantedstock  8295 non-null   float64
 29  plantpatstock    8295 non-null   float64
 30  designpatstock   8295 non-null   float64
 31  plantpat         8295 non-null   float64
 32  designpat        8295 non-null   float64
dtypes: float64(24), int64(7), object(2)
memory usage: 2.1+ MB
```

In [38]:

```
df.columns
```

Out[38]:

```
Index(['country', 'code', 'year', 'eap', 'eca', 'lac', 'mena', 'sha', 'sa',
       'hi', 'pat', 'patepo', 'royal', 'rdexp', 'rdper', 'rdfinabro',
       'rdfinprod', 'rdperfprod', 'rdperfhe', 'rdperfpub', 'lowrdexp',
```

```
       'lowrdfinprod', 'lowrdperfprod', 'y', 'stockpatEPO', 'poptotal',
       'labor', 'rdexpgdp', 'patgrantedstock', 'plantpatstock',
       'designpatstock', 'plantpat', 'designpat'],
      dtype='object')
```

In [36]:

```
df.describe()
```

Out[36]:

	year	eap	eca	lac	mena	sha	sa	
count	8295.000000	8295.000000	8295.000000	8295.000000	8295.000000	8295.000000	8295.000000	8
mean	1981.203014	0.094515	0.195901	0.176974	0.098614	0.150090	0.026522	
std	12.421590	0.292561	0.396917	0.381670	0.298161	0.357182	0.160691	
min	1960.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1970.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	1981.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	1992.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
max	2002.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 31 columns

In [39]:

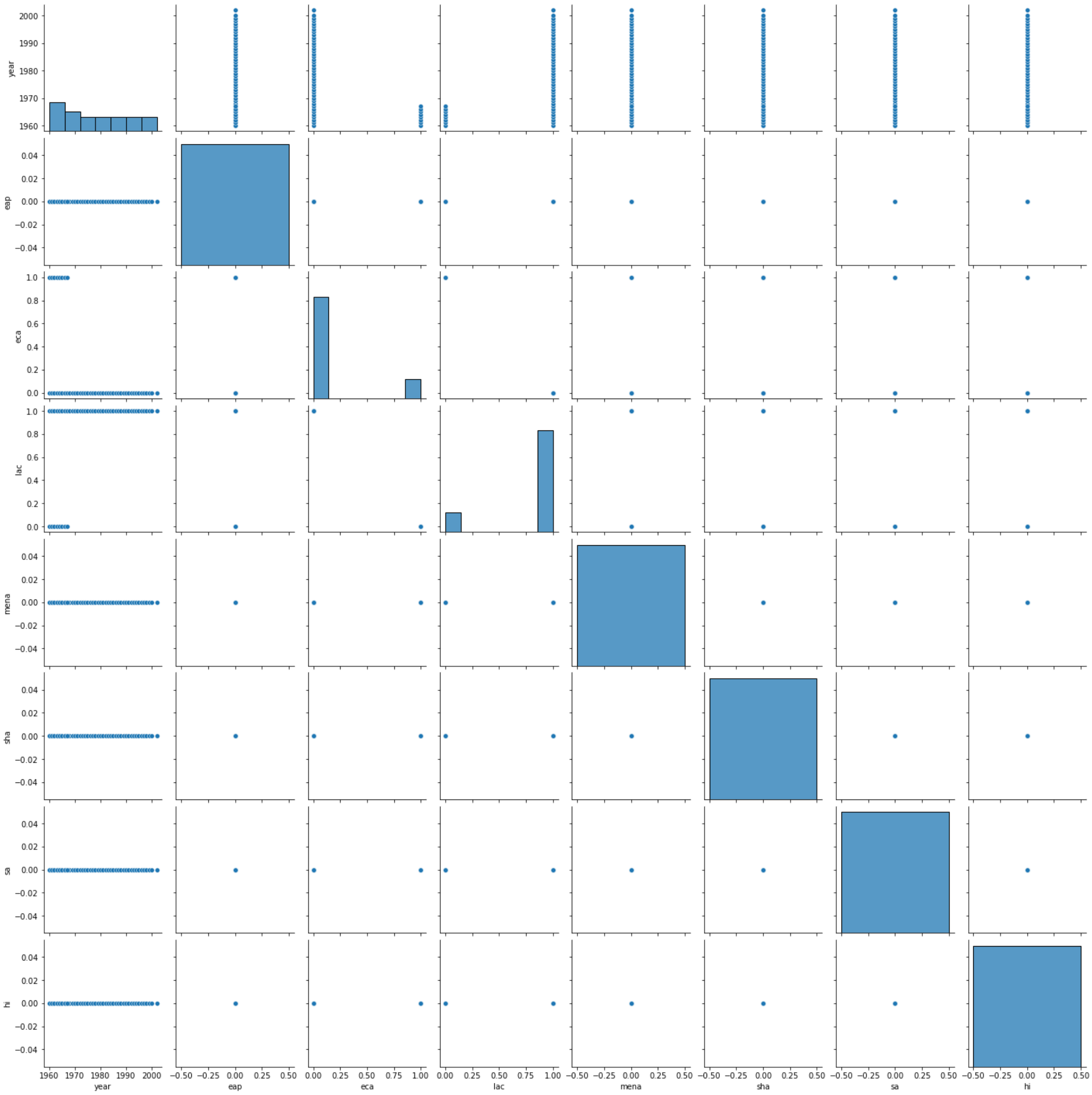
```
df1=df[['year', 'eap', 'eca', 'lac', 'mena', 'sha', 'sa','hi']]
```

EDA AND VISUALIZATION

In [40]:

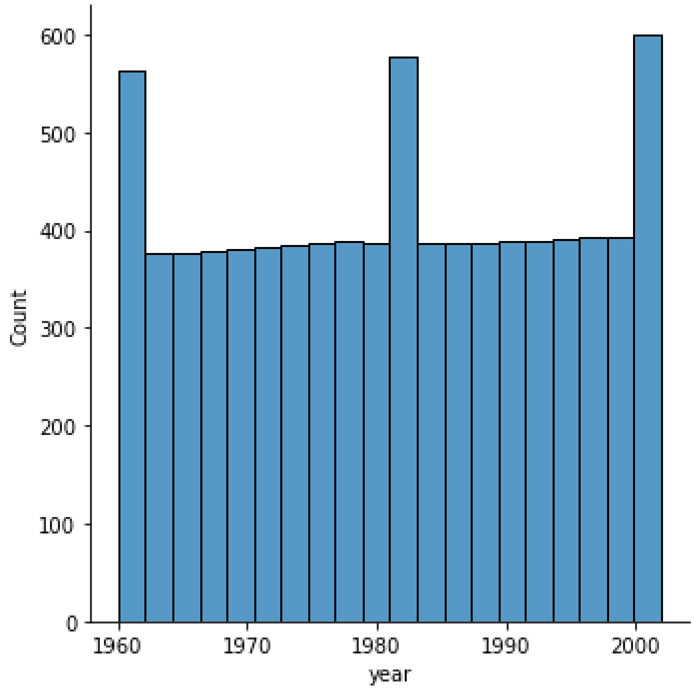
```
sns.pairplot(df1[0:50])
```

Out[40]: <seaborn.axisgrid.PairGrid at 0x2a1696d65b0>



In [44]:
```
sns.displot(df1['year'])
```

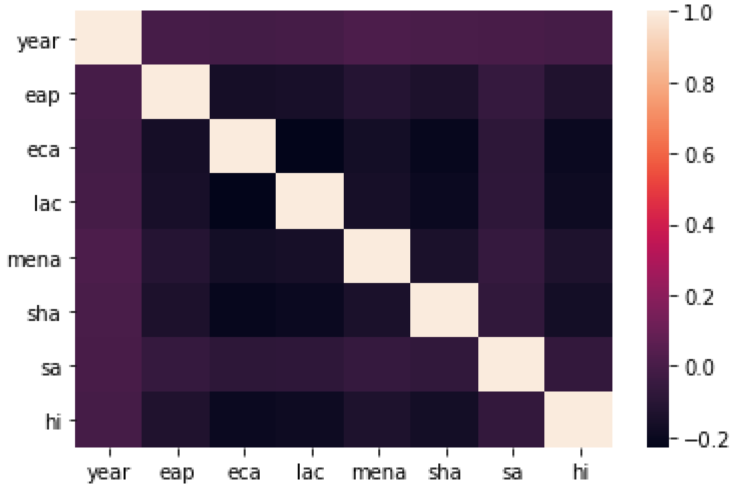
Out[44]: <seaborn.axisgrid.FacetGrid at 0x2a16f3d6a90>



In [42]:

```
sns.heatmap(df1.corr())
```

Out[42]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [45]:

```
x=df[['year', 'eap', 'eca', 'lac', 'mena', 'sha', 'sa']]
y=df['hi']
```

In [46]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Linear Regression

In [47]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

```
lr.fit(x_train,y_train)
```

Out[47]: LinearRegression()

In [48]:
```
lr.intercept_
```

Out[48]: -0.10129916868360311

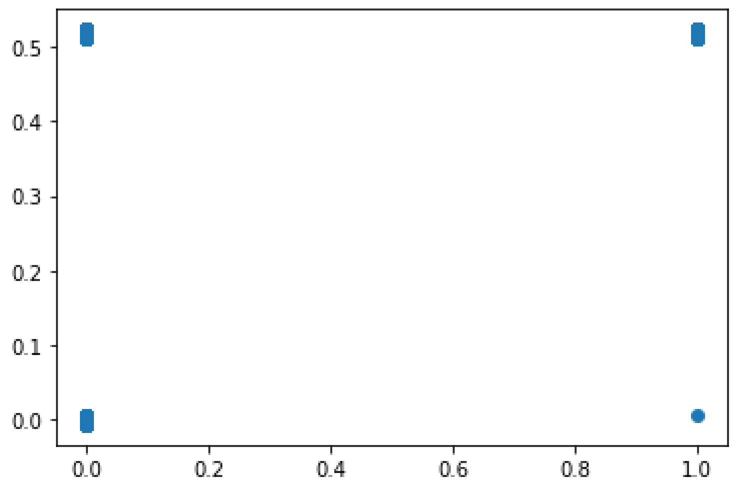
In [49]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[49]:

	Co-efficient
year	0.000312
eap	-0.516784
eca	-0.516617
lac	-0.516767
mena	-0.516982
sha	-0.516911
sa	-0.516918

In [50]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[50]: <matplotlib.collections.PathCollection at 0x2a16fc09d00>



ACCURACY

In [51]:
```
lr.score(x_test,y_test)
```

Out[51]: 0.4613696470505868

In [52]:
```
lr.score(x_train,y_train)
```

Out[52]: 0.442752791624356

Ridge and Lasso

In [53]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [54]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[54]: Ridge(alpha=10)

Accuracy(Ridge)

In [55]:
```
rr.score(x_test,y_test)
```

Out[55]: 0.458186877260473

In [56]:
```
rr.score(x_train,y_train)
```

Out[56]: 0.44143721604040387

In [57]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[57]: Lasso(alpha=10)

In [58]:
```
la.score(x_train,y_train)
```

Out[58]: 0.0

Accuracy(Lasso)

In [59]:
```
la.score(x_test,y_test)
```

Out[59]: -0.00034648662632985605

In [60]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[60]: ElasticNet()

In [61]:
```
en.coef_
```

Out[61]: array([0., -0., -0., -0., -0., -0., -0.])

In [62]:
```
en.intercept_
```
Out[62]: 0.13296589734757147

In [63]:
```
prediction=en.predict(x_test)
```

In [64]:
```
en.score(x_test,y_test)
```
Out[64]: -0.00034648662632985605

Evaluation Metrics

In [65]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```
```
0.23530485566046233
0.12001888817033582
0.3464374231666317
```

Logistic Regression

In [66]:
```
from sklearn.linear_model import LogisticRegression
```

In [67]:
```
feature_matrix=df[['year', 'eap', 'eca', 'lac', 'mena', 'sha', 'sa']]
target_vector=df['hi']
```

In [68]:
```
feature_matrix.shape
```
Out[68]: (8295, 7)

In [69]:
```
target_vector.shape
```
Out[69]: (8295,)

In [70]:
```
from sklearn.preprocessing import StandardScaler
```

In [71]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [72]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```
Out[72]: LogisticRegression(max_iter=10000)

In [74]:
```
observation=[[1,2,3,4,5,6,7]]
```

In [75]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[0.]
```

In [76]:
```
logr.classes_
```

Out[76]:
```
array([0., 1.])
```

In [77]:
```
logr.score(fs,target_vector)
```

Out[77]:
```
0.8772754671488848
```

In [78]:
```
logr.predict_proba(observation)[0][0]
```

Out[78]:
```
1.0
```

In [79]:
```
logr.predict_proba(observation)
```

Out[79]:
```
array([[1.00000000e+00, 8.32677247e-28]])
```

Random Forest

In [80]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [81]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[81]:
```
RandomForestClassifier()
```

In [82]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [83]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy
grid_search.fit(x_train,y_train)
```

Out[83]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [84]:
```
grid_search.best_score_
```

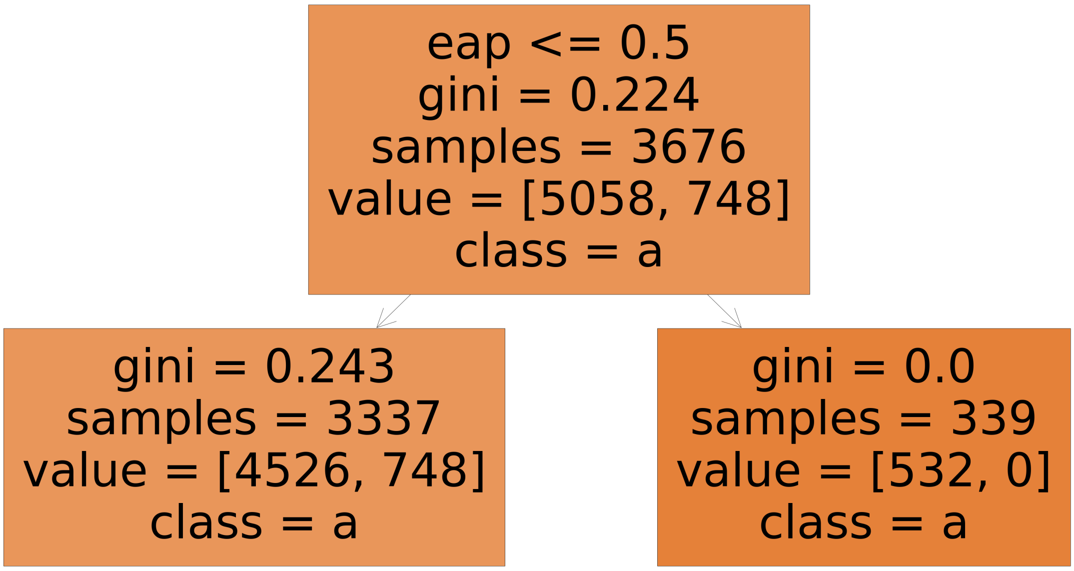
Out[84]: 0.8670341026524285

In [85]:
```
rfc_best=grid_search.best_estimator_
```

In [86]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','
```

Out[86]:
```
[Text(2232.0, 1630.8000000000002, 'eap <= 0.5\ngini = 0.224\nsamples = 3676\nvalue = 
[5058, 748]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'gini = 0.243\nsamples = 3337\nvalue = [4526, 748]
\nclass = a'),
 Text(3348.0, 543.5999999999999, 'gini = 0.0\nsamples = 339\nvalue = [532, 0]\nclass
= a')]
```



Conclusion

Accuracy

Linear Regression

In [87]:
```
lr.score(x_test,y_test)
```

Out[87]: 0.4613696470505868

In [88]:
```
lr.score(x_train,y_train)
```

Out[88]: 0.442752791624356

Lasso

In [89]:
```
la.score(x_test,y_test)
```

Out[89]: -0.00034648662632985605

In [90]:
```
la.score(x_train,y_train)
```

Out[90]: 0.0

Ridge

In [91]:
```
rr.score(x_test,y_test)
```

Out[91]: 0.458186877260473

In [92]:
```
rr.score(x_train,y_train)
```

Out[92]: 0.44143721604040387

Elastic Net

In [93]:
```
en.score(x_test,y_test)
```

Out[93]: -0.00034648662632985605

Logistic Regression

In [95]:
```
logr.score(fs,target_vector)
```

Out[95]: 0.8772754671488848

Random Forest

In [96]:
```
grid_search.best_score_
```

Out[96]: 0.8670341026524285

From the above data, we can conclude that logistic regression is preferrable to other regression types

In [ ]:

In [ ]: